Bolt Beranek and Newman Inc.

Report No. 4354

SPEECH COMPRESSION AND SYNTHESIS

Quarterly Progress Report No. 7
8 December - 7 March 1980

Prepared for:

Advanced Research Projects Agency

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER BBN Report No. 4354	2. GOVT ACCESSION NO. A155 762	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SPEECH COMPRESSION AND SYNTHESIS		5. TYPE OF REPORT & PERIOD COVERED Quarterly Progress Report 8 Dec. - 7 March 1980
		6. PERFORMING ORG. REPORT NUMBER Report No. 4354
7. AUTHOR(s) Michael Berouti John Klovstad John Makhoul Richard Schwartz John Sorensen		8. CONTRACT OR GRANT NUMBER(s) F19628-78-C-0136
9. PERFORMING ORGANIZATION NAME AND ADDRESS		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE March 1980
		13. NUMBER OF PAGES 39
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Deputy for Electronic Technology (RADC/EEV) Hanscom AFB, MA 01731 Contract Monitor: Mr. Caldwell P. Smith		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce, for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 3515.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Speech synthesis, phonetic synthesis, diphone, LPC synthesis, vocoder, speech compression, linear prediction, voice-excited coder, high-frequency regeneration, spectral duplication, phonetic vocoder, spectral template, phoneme recognition.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This document reports on work toward a very low rate phonetic vocoder and a multirate speech compression system. The phonetic vocoder consists of a phonetic recognizer based on a trained diphone network, and a natural phonetic synthesizer which also uses diphone templates as a model for speech. This quarter several new diphones were added to the data base. There are currently 2845 diphones. We tested the phonetic recognition program, and made improvements to improve its speed and performance.

Report No. 4354

SPEECH COMPRESSION AND SYNTHESIS

Quarterly Progress Report No. 7
3 December - 7 March 1980

Prepared by:

Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, Massachusetts 02138

Prepared for:

Advanced Research Projects Agency

TABLE OF CONTENTS

1. SUMMARY

In this Quarterly Progress Report, we present our work performed during the period December 8, 1979 to March 7, 1980.

1.1 Introduction

The work in the last quarter was in the areas of natural phonetic synthesis, phonetic recognition, and multirate speech compression. The recognition and synthesis programs will operate together as a very low rate phonetic vocoder. Below is a summary of the accomplishments in each of the three areas. Details are presented in Sections 2, 3 and 4.

1.2 Phonetic Synthesis

The exhaustive testing of the diphone templates has been proceeding according to schedule. We have also added 50 new diphones to allow more natural synthesis of flapped /t/. Rules were added to the synthesis program that allow the insertion of glottal stops and glottal onsets.

In order to use the diphone synthesis program as the final stage of the MIT text-to-speech system, an interface program was written to convert the intermediate phonetic output of the MIT text-to-phone program into a form suitable for the diphone synthesis program. This will also allow the comparison of the phonetic synthesis-by-rule program with the diphone synthesis program.

1.3 Phonetic Recognition

During this quarter improvements were made to the phonetic recognition programs which affect its performance, efficiency, and interactive capabilities.

We added statistical programs to permit the continuing refinement of the mean and standard deviation parameters now stored at every network node. We added training code which permitted us to take statistics based on the result of a match or to add new paths or path segments to the network from the current input frames. We generalized the matching algorithm so that theory merging could be done at every network node. We made it easier for the user to realize the alignment desired by extending the forced alignment to include time constraints. All program control, as well as the setting of many control parameters, was made directly accessible via a thoroughly interactive command loop.

Our future work is discussed as it appears in light of these recent improvements.

1.4 Multirate Speech Coding

During this quarter we tested the complete adaptive predictive coder as an embedded-code multirate coding system at data rates ranging from 6.4 kb/s to 16 kb/s. We investigated three embedded coding techniques and improved our high-frequency regeneration method. Our results show that the baseband coder approach yields generally good quality speech in the range 6.4 - 9.6 kb/s. The details of our work are reported in Section 4.

2. SYNTHESIS

The major effort this quarter in synthesis has been directed toward exhaustive testing of the diphone data base. We have also recorded and digitized another 50 diphones to account for flapped /t/ [DX] before and after each vowel.

Two substantial programming additions were made. The first allows the synthesis of glottal stops and glottal onsets. The second was an interface between the MIT text-to-speech system and the BBN diphone synthesis program.

The data base additions are described in Section 2.1, diphone testing is discussed in Section 2.2, and programming changes are outlined in Section 2.3.

2.1 Diphone Data Base.

Now that the initial transcription (labelling) of the diphone data base is complete (see QPR No. 6), we have turned our attention to the testing and "tuning" of the data base. This testing (described below) has resulted in several changes to the structure of the data base.

We have, first of all, determined that we need to include the phoneme "flap" (as in "bu<u>tt</u>er") in all possible vowel contexts. We are currently extending the data base to include all possible English combinations of [Vowel DX] and [DX Vowel]. This should increase the naturalness of transitions at word boundaries such as: "You bet I will") where the /t/ is typically flapped.

One issue associated with the synthesis of flaps is their segmental duration; flaps are usually 10 to 20 ms in length. Therefore, some effort will have to be spent on deciding how to synthesize flaps from their stored diphone templates. For example, we have defined certain "elastic" and "inelastic" regions in each diphone, referring to the degree to which these regions can be stretched and/or shrunk during time-warping of the diphone templates to produce synthetic speech. We must determine how this scheme should be applied to phonemes of very short duration, such as flaps.

Another change to the data base concerns the glottal stop phoneme, [Q]. Instead of using data extracted from real speech, we have determined that glottal stops should be generated by rule. The major acoustic attribute of a glottal stop is a lowered pitch contour. An experiment reported in [1] indicated that this pitch lowering occurs mainly over the first 50 ms of the vowel, and has a magnitude of about 30-50 Hz. The details of how this rule is implemented in the synthesis program are described in Section 2.3.1.

2.2 Diphone Testing

The exhaustive testing of all the diphones in the data base is progressing according to schedule. This testing consists of using each diphone in an automatically generated nonsense utterance. Incorrect labelling of the original diphones can usually be detected by closely examining any nonsense utterance that doesn´t sound natural. We have found that only about 5% of the diphones need some correction. We expect that testing will be completed by the beginning of June.

One result of this exhaustive testing is the occasional modification of our rules for the placement of phoneme boundaries in the diphone utterances. For instance, we found that the boundary between a strident fricative (such as [S] or [SH]) and a following vowel needed to be placed one frame later into the vowel order to avoid a pop in the synthesis due to misalignment between the excitation and the spectrum.

As mentioned in previous QPR´s, the diphones must also be tested by synthesizing complete, natural sentences. As the exhaustive testing nears completion, time spent on this second phase of testing will increase. By testing in this mode we expect we may discover a need for context-specific diphones that were not anticipated in the original data base.

2.3 Program Changes

There were two additions made to the diphone synthesis programs. The first addition allowed the synthesis of glottal onsets and glottal stops which were recorded explicitly for the data base. The second addition was a program to interface between the MITALK text-to-speech system and our diphone synthesis program.

2.3.1 Glottal Stops by Rule

Based on acoustic evidence from Sorensen and Cooper [1], as well as our own informal observations, we have implemented a rule for synthesizing glottal stops. A glottal stop manifests itself mostly as a drop in pitch and energy. The drop in these parameters is very abrupt, and the rise back to normal levels can be approximated by a linear function of time.

In our program, the onset of the glottal stop is marked by a drop in pitch of 50 Hz, and a drop in energy of 5 dB from the previous vowel. These values continue for half the duration of the glottal stop phoneme. The pitch and energy are increased linearly during the second half of the glottal stop phoneme, so that they reach the level specified for the next vowel when the interval is finished.

We are experimenting with this rule to fine-tune the parameters, but are quite pleased with the natural sounding glottal stops that have been produced thus far.

2.3.2 Text-to-Speech

The MITALK text-to-speech system takes typed text as input, and produces a digital speech waveform as output. The system is organized into several programs, each of which performs some part of the transformation from text to speech. The communication between programs is accomplished through text files. The last program (synthesis-by-rule) takes as input a list of phonemes, durations, pitch targets and stress levels and produces a speech waveform. This data is similar to the input required by our diphone synthesis program.

A conversion program was written to translate the synthesis-by-rule text file so that it can be read by our diphone synthesis program. The translation is necessitated by four types of differences between the conventions for labelling used by the two programs.

First, several phoneme names are different. For example, the symbols [NG] in the MIT program and the symbol [NX] in the diphone synthesis program both refer to the velar nasal (as in "si<u>ng</u>"). This translation requires only a simple lookup. In some cases the mapping from one symbol set to the other depends on the adjacent phonemes. For instance the diphthong [YU] is changed to the sequence [Y-UW] when followed by another vowel.

The second change involves the definition of plosive regions. In the MIT system, plosives are specified by a single phoneme [P T K B D G]. However, the plosive actually consists of a silence region followed by a burst and (if unvoiced) a period of aspiration. These regions are predicted by the MIT synthesis program. The diphone synthesis program treats plosives as two phones: a silence and a burst-aspiration. The durations of these two phones is determined by the conversion program using rules similar to those used by the MIT program. The rules take into account the plosive, the surrounding phonemes, and the stress pattern.

The third change is concerned with the location of phoneme boundaries. For some phoneme pairs (e.g. strident fricatives followed by vowels), the conventions don't match, and so some of the phoneme durations are modified.

The fourth difference involves a scale factor in the units for duration and pitch. This is performed by simple division by the scale factor.

The resulting text file can be read into the diphone synthesis program using a new file input option programmed for this purpose. The combined MIT-BBN system has been used to synthesize several sentences.

We have not yet performed a careful comparison between the two phoneme-to-speech programs. However, limited informal listening indicates that the diphone synthesis program produces more realistic articulation. On the other hand, the MIT synthesis-by-rule program seems - at present - to produce slightly smoother speech than the diphone synthesizer. A detailed comparison will require more time.

2.3.3 Future Work

The diphone testing and sentence testing will continue during the remainder of the contract. At this point, we believe that incorrect labelling does not account for many of the remaining problems. We will be working on these problems by trying new instances of some diphones, and introducing modifications to the diphone concatenation program. Although we expect continued improvement of diphone synthesized speech, the speech already produced is highly intelligible and fairly natural sounding.

3. PHONETIC RECOGNITION

During this past quarter many additions and improvements were made to the phonetic recognition programs. These are discussed in Section 3.1. The anticipated direction of future work is discussed in Section 3.2.

3.1 Phonetic Recognition Programs

In this section the additions and improvements to the phonetic recognition programs are discussed. There are two fundamentally distinct programs which we will talk about in this section. One is the compiler, which takes a text file of diphone descriptions (in essentially the same format as the input used by the synthesis compiler) and produces a network. This compiler has the capability to make an incremental addition to an already existent compiled network. It is this second capability which will permit us to train the network and later add entirely new diphones (if necessary) but retain the training on the original network. The second program is the matcher. This is the program that actually does the phonetic recognition. It reads in a compiled network (trained or untrained) and uses its paths when trying to determine the best diphone sequence. The matcher permits the user to specify a desired result (in terms of a sequence of phonemes and optional time limits) and to train on an actual alignment (correspondence between network paths and input frames) once the match has been completed. This training consists of collecting statistics (which can effect both the spectral and duration scoring) or adding new paths to the current network.

Each of the major changes which have been made are described. The motivation for the change is presented and the necessary programming to implement the change is mentioned.

3.1.1 Statistical Scoring (Matcher and Compiler)

Perhaps the most important change was to generalize the spectral scoring capability. As of last quarter the spectral score used was the weighted Euclidean distance between a network node and its corresponding input frame whose spectral "coordinates" are Log Area Ratios. The weighting was independent of the particular network node however. Therefore, in order to make this weighting dependent on the particular network node being scored, weighting information was added to each network node, in the form of standard deviations. This change required a modification to both the compiler and the matcher. It required a modification to the compiler because the format of the spectral information stored at each network node had to be extended to include room for the standard deviation of each parameter. (This extended spectral information is equivalent to specifying a diagonal covariance matrix at each network node.) The compiler stores the same set of standard deviations (one for each of the Log Area Ratio parameters) in every node of the network. (The matcher, however, can change both the means and standard deviations of any network node during statistical training.) The matcher also required a change for the same reason (different format for spectral information) but also required a change in the spectral scoring code so that the new "local variance" information is used. Both of these changes were in anticipation of the statistical programs discussed in Section 3.1.4 but did not by themselves alter the performance of the phonetic recognition.

One of the immediate implications of this change, however, was the growing realization that the anticipated memory space problem might occur sooner than we had suspected.

3.1.2 Forced Alignment With Time Constraints

It had been possible to specify a forced alignment as a sequence of phonemes. It was not possible to control where these specified phonemes would be matched with the input. Sometimes this didn't matter since the best scoring alignment closely agreed with the manual transcription. The alignment capabilities were extended to permit the user to have the option of specifying a time range (before, after, at, between) for the beginning of each phoneme in order to control the matching alignment. Occasionally we thought it advisable to leave the particular phoneme unspecified (particularly whenever there was some question as to the accuracy of the labelling) so a phoneme "ANY" was permitted in the alignment specification. Finally, since the sentences frequently ended with an indefinite period of silence, the alignment specification was extended to automatically permit an arbitrarily long silence at the end of any sentence.

The implementation of these changes directly affected only the matcher program. It indirectly affected both programs in that the necessity for permitting merging of competing theories at every node in the network became apparent because of this. This theory merging, which is discussed in the next section, would have been desirable had the alignment specification changes not been made. However, the implementation of time constraints on the forced alignment made this need much more noticeable than it might otherwise have been. Consequently, an implementation allowing more frequent merging of competing theories was strongly encouraged.

3.1.3 General Theory Merging

An alignment defines a specific correspondence between frames in the input and nodes in the network. Each theory consists of a unique alignment and an associated score. If two or more partial theories arrive at the same network node at the same time, i.e., each theory is attempting to score this node with the same input frame, only the highest scoring theory needs to be kept. Originally it was felt that merging theories at diphone boundaries would be satisfactory. The matcher seemed to function well and did not generally "discover" better scoring paths if the theory stack size were doubled or tripled. When the time constraints were included as part of the forced alignment code we noticed that frequently the entire theory queue would be depleted and the matcher would fail to find a match that was consistent with the requested alignment constraints. The history of the matching process was carefully examined by looking at the theories on the stack at each point in the input. Although theory merging was being done properly at each of the diphone boundaries, the number of theories actually on the theory stack was several times as large as it would have been if theory merging had been done at every single node in the network. While simply increasing the stack size would have given us the result we desired (an alignment consistent with the time constraints) we recognized that it was time for the more general theory merging capabilities to be added.

This change affected both the compiler and the matcher since a change in the network was necessary. Rather than describing the structure of the network it will only be noted here that this change requires more memory per network node than was previously required. In order to take advantage of this extended network structure, the matcher had to be changed to detect and merge "comparable" theories.

This change was a very significant one for two reasons: First, the new strategy for theory merging permitted a much smaller theory stack and faster operation for equivalent performance. Second, the additional memory requirements made it very clear that we were about to run out of memory space.

This precipitated yet another round of changes in order to reduce the memory space required by the compiled network. It appears that a "reasonable" amount of training and network modification will now be possible before we run out of memory space again.

3.1.4 Training

Given an alignment, that is, a correspondence between input frames and network nodes, the user can train the network in many ways:

a) Update the Statistics at each network node.

The user is permitted to collect statistics on any region of an alignment that results from a match. Once a match is made and the resulting alignment printed, the correspondence between frames in the input and nodes in the network can be seen. The user specifies that statistics are to be collected over a certain region of the input. The correspondence between input frames and network nodes is used to determine which network nodes are to be changed. Specifically, the spectral statistics (means and variances of each parameter) of a network node are updated whenever an input frame is specified to which it corresponds in the matched alignment. When several consecutive input frames have all been aligned with the same network node, each of them is used as a single sample to update the network spectral statistics.

The duration statistics at each network node used in the match alignment is updated by one sample (for each distinct time it is used in the statistics command) which corresponds to the number of consecutive input frames to which it is matched in this alignment.

It is important to note that statistics can only be taken if a match has already been made and that the portion of alignment specified in the statistics command is sufficient to completely determine what network nodes are to be updated.

b) Add a new diphone from the input frames.

Since the network compiler builds its network by including entire diphones as one path between phoneme nodes, we included a similar capability in the matcher. Each of the diphones reported in the match alignment are numbered. The user indicates which sequence of input frames are to be used as the diphone path by specifying the initial and final input frames. He indicates which diphone this path is to define by indicating its number in the match alignment. The user must indicate which input frame is to correspond to the phoneme boundary. The very same variable frame rate (VFR) algorithm used by the compiler is used by the matcher to reduce (possibly) the actual number of nodes stored in the network.

Notice that this procedure permits only the specification of alternative diphone samples. The network compiler would have to be used if a completely new diphone were to be added to the network.

c) Add a new branch within a diphone.

The user can specify that a portion of the input be used as a path segment to branch past part of the already existent network paths as long as the specified path is to be completely contained within one diphone. The added path segment is not allowed to cross a diphone boundary. The input path is specified by its initial and final frame numbers. The network nodes that it is to bypass is implied by those network nodes that correspond to the specified input region in the current match alignment. If the path would bypass the phonemes boundary (somewhere in the middle of the diphone) the user must indicate which input frame is to be marked as the boundary. The matcher also uses the VFR algorithm for the compression of these input paths before their inclusion in the network.

3.1.5 Skipping Network Nodes

After all of the above mentioned changes had been implemented and training had been started, it was noted that some diphones compiled into the network were longer than the instances in the input which they were expected to match. Previously, the matching allowed more than one input frame to match a single network node (self loop), but required at least one input frame for each node. Thus, whenever the network diphone contained more nodes than there were frames in the corresponding region of the input, the correct alignment was impossible. This problem had been anticipated and been largely dealt with by using the UFR algorithm on every diphone included in the network. Very often, however, the alignment could be realized locally and statistics and/or new paths added to the network. Whenever new paths were added to the network in this way, the fact that the paths were taken from the input and the use of the VFR algorithm frequently permitted subsequent matches to match the global alignment much more accurately. The problem with this procedure, however, was that it was very tedious, since several matches would sometimes be required before the desired alignment could be realized.

It was primarily in response to this problem, the tediousness of training, that the capabilities of the matcher were changed so as to automatically permit a single input frame to simultaneously match two consecutive network nodes. Although this addition violates the spirit of the scoring philosophy as presented in the last QPR, it was added with the understanding that alignments that do not require its use are to be preferred to those that do. This additional matching capability was added on a flag and is going to be used primarily to enable much quicker attainment of the desired alignments. Once the desired alignment has been established, the training capabilities of the matcher can alter the network with new paths that do not require such a skipping of network nodes. Even when the flag is on and such a match is made, network frames are not really skipped as such, since both frames are scored against the same input frame.

It is possible that we will find it convenient to permit this kind of matching routinely, since the current indications are that scoring a single input frame with consecutive network nodes only infrequently results in better scoring alignments (and then only when necessary) when the matcher is permitted to run without any forced alignment.

3.1.6 Interactive Code

Since the matcher code already depended on the specification of numerous parameters and the different kinds of operations in addition to a match was growing rapidly, we included code that made the matcher operation interactive. At the top level the user can request a match, change an alignment specification (if one already exists) or any of a number of controlling parameters (Stack size, default duration scores etc.), train on an existing alignment, write out the current network (statistics and new paths are included) or quit. Each of the change and/or train commands at this top level leads to a new level of command structure in which several commands are possible.

3.2 Future Work

The purpose of including this section is to state as clearly as we can where we see our research time going in the remaining portion of this project.

3.2.1 Ongoing Debug and Program Improvements

First, we will have to continue the process of tracking down each newly discovered bug as it becomes apparent during the ongoing testing and training of the recognition programs. At this time it is very difficult to see what improvements will be necessary since practically all of the original ideas have now been implemented and the training is just beginning. From our past experience we do anticipate that some changes will be necessary although we hope that most of the remaining time can be spent on the training of the network.

3.2.2 Training

Since the training of the network is the most important remaining part of the work to be done (assuming that no new code needs to be written) we expect to spend a great deal of time during the next quarter running the system on sentences on which we want to train.

Currently we intend to train some subset of the network first (rather than to try to get samples of every diphone) to investigate its behavior as a system. By first training on those diphones which occur most frequently in natural English we expect to improve the overall system performance rapidly. Our subsequent training on less frequent diphones is also expected to improve the system performance but at a substantially reduced rate.

4. MULTIRATE CODING

4.1 Introduction

During this quarter we have tested the complete system as a multirate coder with the constraint of embedded coding. We concentrated mainly on two aspects of this project: (i) embedded coding techniques, and (ii) high-frequency regeneration methods. In the following, we shall report only on those two aspects of the system, since we have discussed the other details of the transform coder in previous reports.

4.2 The Multirate Feature of the System

At each frame, the transmitter transmits a block of bits which is divided into two major parts. The first part contains the bits representing the system parameters and the second part contains the bits representing the discrete cosine transform (DCT) components. One such block of bits is shown in Fig. 4.1. The transmitter always operates at the same fixed bit rate; it is the channel's function to discard some of the transmitted bits to alleviate traffic congestion. Thus, the maximum data rate of 16,000 b/s is achieved when the DCT of the fullband residual is transmitted. The minimum data rate achievable by the system takes place when all the codes representing the DCT components

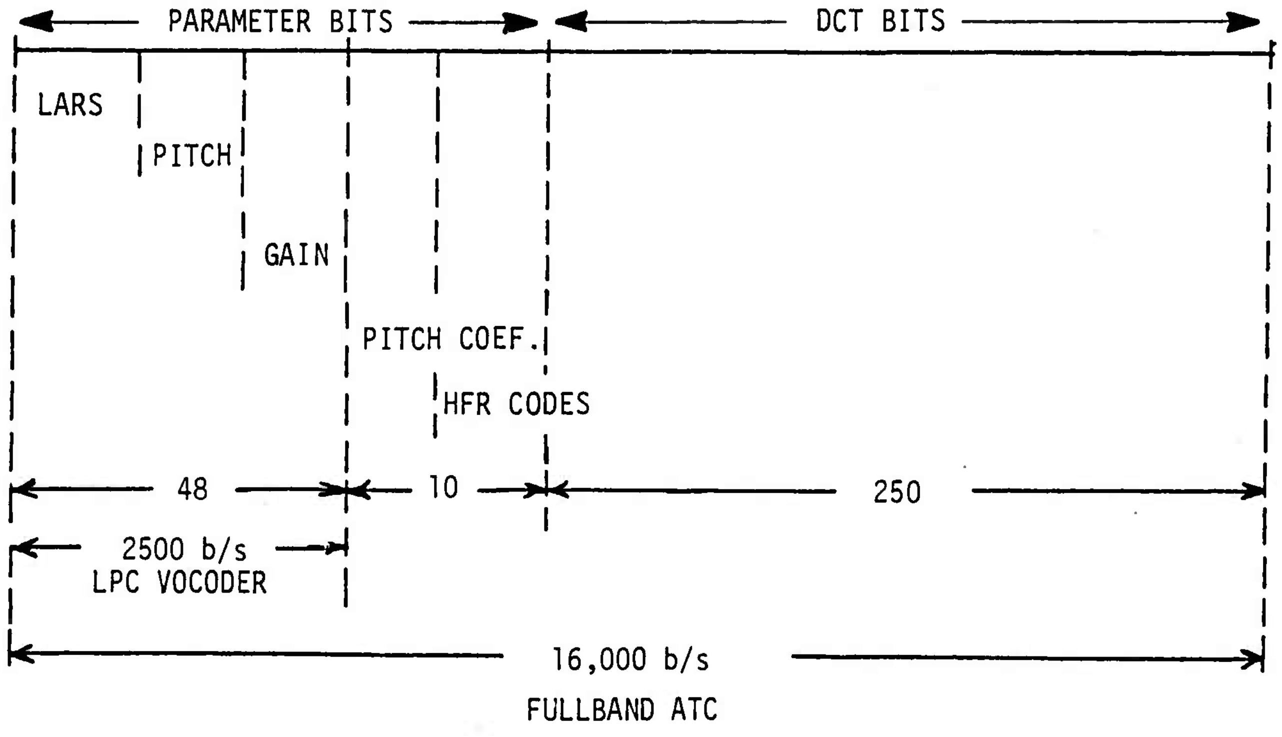


Fig. 4.1 One block of bits per frame illustrating the embedded-code multirate feature of the system.

are suppressed by the channel and only the system parameters are transmitted. In such a case, the receiver becomes identical to that of a narrowband pitch-excited LPC vocoder operating at 2500 b/s.

Intermediate data rates are achieved by stripping off bits from the high data rate system. Stripping off bits results in the suppression of low-amplitude frequency components and/or high-frequency components. This aspect of the system will be explained in Section 4.1. Here, we point out that, at the intermediate data rates, the receiver regenerates the missing frequency components to restore the fullband DCT. An inverse DCT yields the time-domain residual waveform which is used as input to the all-pole LP synthesis filter.

It is worth mentioning here that the transmitter itself can strip off bits prior to transmission in the same manner as is done by the channel. Thus, when a system is first turned on, the initial bit rate need not be high and traffic congestion can be avoided. Note that the receiver does not need to know where in the system the bits are discarded.

4.3 Embedded Coding

As mentioned above, the transmitter transmits at each frame one block of bits. It is assumed that the channel strips off bits starting at the end of the block, thereby discarding bits that represent DCT components. The codes representing the DCT components are arranged in a certain order prior to transmission. This ordering determines which bits get discarded first. To study the tradeoff between the number of transmitted bits, the quantization accuracy, and the number of received frequency components, we investigated three ordering techniques. In all three techniques, to be described below, we assume that the receiver decodes the system parameters and performs the bit allocation as was done at the transmitter. This is standard practice in adaptive transform coding. In addition, we require that the receiver know how many bits are received each frame so that it knows where the next frame begins. This last piece of information is passed along by the channel itself.

The first bit-ordering technique we investigated is the simplest: the codes are arranged by order of increasing frequency. When the channel strips off bits from the end of each block, the high-frequency components are discarded first. The remaining codes represent a low frequency portion of the total bandwidth referred to as a baseband. As in a baseband coder, the receiver regenerates the missing high-frequency components. We use the method of high-frequency regeneration (HFR) by spectral duplication, which is explained in Section 4.4.

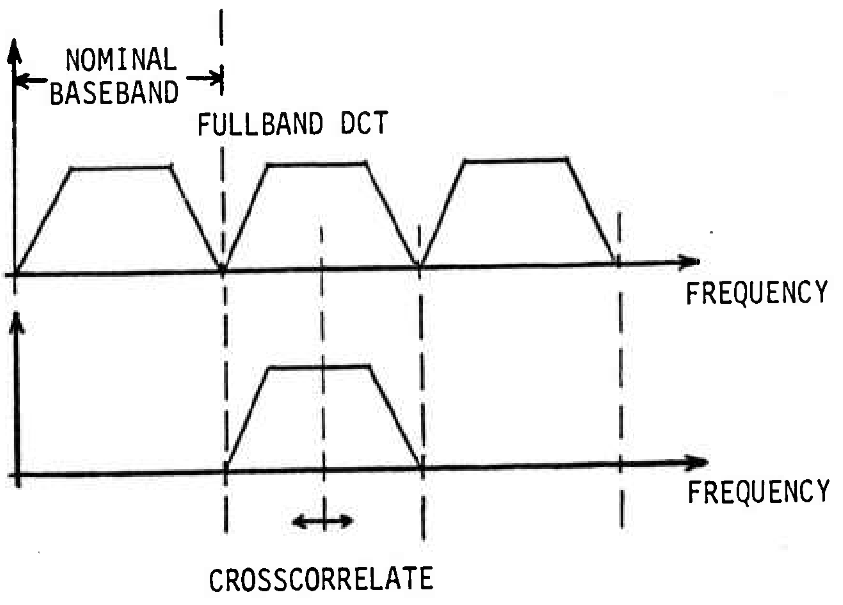
In the second ordering technique, the DCT codes are broken down into individual bits. The bits are then grouped together by order of significance, with the most significant bits in the first group and the least significant bits in the last group. To explain this grouping method, let us assume that the bit allocation produces codelengths between 1 and 5 bits. Therefore, the receiver expects to see 5 groups of bits. The first group contains the most significant bit of all 5-bit codes (in order of increasing frequency). The second group of bits contains the next most significant bit of all 5-bit codes and the most significant bit of all 4-bit codes. And so on, all the way to the last (5th) group of bits which contains the least significant bit of all the codes. When the channel strips off bits, the first bits to be discarded are the least significant bits of each transmitted code, resulting in decreased quantization accuracy. For example, a DCT component originally coded into 3 bits will be decoded by means of the 2-bit decoding table if its least significant bit has been dropped.

In the third ordering technique, the codes are grouped together according to their length. Thus, the receiver expects to see 5 groups of codes, with the first group containing all 5-bit codes, the second group all 4-bit codes, and so on, with the last group containing all 1-bit codes.

The second and third ordering techniques described above are more complex than the first, since they require arranging the data in the order of decreasing codelength. Informal listening tests have shown that the quality of the reconstructed speech when using the baseband-coder approach (the first ordering technique) is superior to that obtained by the second and third techniques. In general, our experience has been that the details of the low-frequency components of speech are perceptually more important than those at high frequencies. Thus, the task is to find a good compromise, at a given bit rate, between baseband width and quantization accuracy. At present, we feel that the first technique is giving the best overall speech quality for bit rates in the range 6.4 to 9.6 kb/s. We have not compared the three techniques at bit-rates above 9.6 kb/s. However, it is worthwhile pointing out here that we are seeking one single technique that will perform uniformly well over the whole range of data rates, because we are excluding the possibility of changing the coding algorithm while the system is in operation.

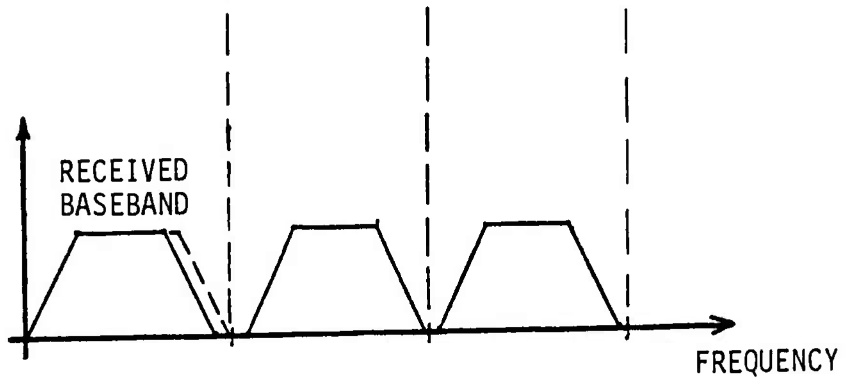
4.4 High Frequency Regeneration

We have presented elsewhere [2] new HFR methods based on duplication of the baseband spectrum. In particular, we presented time-domain systems that perform spectral duplication by spectral folding and by spectral translation. We also present a frequency-domain system [3] that performs HFR by spectral translation of the baseband. The principle of the method is based on the fact that, in the adaptive transform baseband coder, the baseband DCT components can be easily duplicated at higher frequencies to obtain the fullband excitation signal. Our present HFR method aims at duplicating, as closely as possible, the original fullband DCT of the residual, while accommodating the variable baseband width aspect of the present multirate system. We now explain the method with the help of Fig. 4.2. The transmitter assumes a (fixed) nominal baseband width of 1000 Hz. Thus, the simplest spectral translation method would be to duplicate the region from 0 to 1000 Hz onto the regions from 1000 to 2000 Hz and from 2000 to 3000 Hz. In addition, to lock the high-frequency interval into place, by exploiting the quasi-harmonic structure of the speech spectrum, we shift the baseband around its nominal position and correlate it with the corresponding original DCT components that are in the region 1000 to 2000 Hz. The cross-correlation is done at the transmitter where the original DCT of the fullband residual is available. The same process is repeated for the next frequency band. Short lags from -3 to +4 spectral points are considered. (The total bandwidth is 128 points.) The optimal location is then chosen to be at the positive maximum value of the cross-correlation. Thus, we require an additional 3 bits of



A. At the transmitter

1. Choose point of maximum correlation.
2. Transmit 3-bit HFR codes



B. At the receiver

1. Translate rec ived baseband
2. Fill gaps

Fig. 4.2 High frequency regeneration by spectral duplication

side information for each of the two high-frequency bands. The additional 6 bits are transmitted along with the system parameters (HFR codes).

At the receiver, the decoded baseband is translated up, starting at 1000 Hz (and 2000 Hz for the next band) and is moved further by a small amount as indicated by the 3-bit HFR code.

In practice, there are three deviations from the simple algorithm described above. The first is that the first few DCT components, starting at d.c. and up to half the pitch frequency, are not duplicated onto the high-frequency bands, nor are they considered in the correlation method described above. Second, we found that spectral flattening of the baseband at the receiver prior to HFR improves the speech quality. The third deviation of the algorithm is due to the fact that the received baseband width is seldom equal to 1000 Hz. In fact, it varies from frame to frame, because the received number of bits (and therefore number of DCT components) is set by the channel. We have devised certain modifications to the method to deal with that problem appropriately. For example, following the HFR process described above, there can be gaps in the regenerated fullband DCT, as illustrated in Fig. 4.2. To fill such gaps, we translate the received baseband in such a manner that its center coincides with the middle of the gap. We then shift the baseband and correlate it with the reconstructed DCT spectrum. The point of maximum correlation indicates a best match between the baseband and the DCT components around the gap. Therefore, at that shift, we copy the appropriate baseband components onto the gap.

4.5 Results

We performed a large number of experiments to study the tradeoffs between all the interacting aspects of the system described above. We summarize our results briefly below. The basic system is governed by the already existing ARPANET LPC vocoder operating with a frame size of 19.2 ms, i.e., 128 points, and transmitting 9 log-area-ratios each frame coded into 5,5,5,4,4,4,3,3, and 3 bits, respectively. Together with pitch and gain, the system parameters require a total of 48 bits. Thus, the basic LPC vocoder operates at a bit-rate of 2500 b/s. In addition, we require 4 bits for the pitch tap and 6 bits for the HFR codes, bringing the side information to 58 bits per frame. The remaining bits, which determine the total bit rate, are used to code the DCT coefficients. We investigated the following aspects of the system.

a. Initial Quantization Accuracy. To study the interaction between quantization accuracy and the number of received DCT components at a given data rate, we coded the fullband DCT at an average of 2, 2.25, 2.5, and 3 bits per sample. In our experiments, there was no maximum limit on the bit rate for the fullband case, although, in practice, the system will be operated at 9.6 kb/s or below.

b. Noise Shaping. We used various values of the noise shaping parameter γ [see previous QPR] ranging between 0 and 1. For γ closer to 1, the available bits are spread more evenly in the frequency range, resulting in a larger number of received DCT components at a given bit rate, at the expense of coarser quantization in the low-frequency region for voiced sounds.

c. Embedded Coding. We simulated the three bit ordering techniques described in Section 4.3 and evaluated informally the speech quality obtained with each. For each technique, we optimized the system in terms of the total fullband rate and the value of γ as in (a) and (b) above.

d. High-Frequency Regeneration. As described in Section 4.4, the transmitter assumes a nominal baseband width for which it computes the HFR codes. We investigated the use of 3-bit and 2-bit HFR codes, and the use of a nominal baseband width of 800, 900 and 1000 Hz.

For each choice of the above described parameter settings we tested the system at 9.6 kb/s, 7.2 kb/s and 6.4 kb/s, although, in principle, the data rate can be set by the channel to an arbitrary value. All tests were done with 5 male and 5 female sentences. Our present choice of a good compromise system operating uniformly well over the desired data rates is one where we code the fullband DCT of the residual at an average of 1.95 bits per sample, i.e., where the maximum bit rate is 16 kb/s. The value of γ is 0.9, and the embedded coding technique is the first (baseband coding), with a nominal baseband width of 1000 Hz and 3-bit HFR codes. The data rates of 9.6, 7.2 and 6.4 kb/s are achieved by keeping at each frame 124, 80, and 65 bits, respectively, out of the maximum total of 250 bits. The average width of the received baseband for the three cases is 1400, 870, and 670 Hz, respectively.

At present, we feel that the above described system is providing us with very good speech quality at 9.6 kb/s, good speech quality at 7.2 kb/s, and reasonable quality at 6.4 kb/s. The problem at bit rates of 6.4 kb/s or below is that the received baseband becomes too narrow, which results in appreciable roughness in the coded speech and some "thuds." Also noticeable at 6.4 kb/s, especially for female voices, is the reverberant quality of the reconstructed speech.

5. REFERENCES

1. J. Sorensen and W. Cooper, "Syntactic Coding of Fundamental Frequency in Speech Production," Perception and Production of Fluent Speech, Lawrence Erlbaum Associates, Publishers, pp. 399-440, 1980.

2. J. Makhoul and M. Berouti, "High-Frequency Regeneration in Speech Coding Systems," Int. Conf. Acoustics, Speech, and Signal Processing, Washington, D.C., pp. 428-431, April 1979.

3. M. Berouti and J. Makhoul, "An Adaptive-Transform Baseband Coder," Proceedings of the 97th Meeting of the Acoustical Society of America, paper MM10, pp. 377-380, June 1979.